A UNIFYING FRAMEWORK FOR IMAGE INPAINTING

By

Aurélie Bugeau

Marcelo Bertalmío

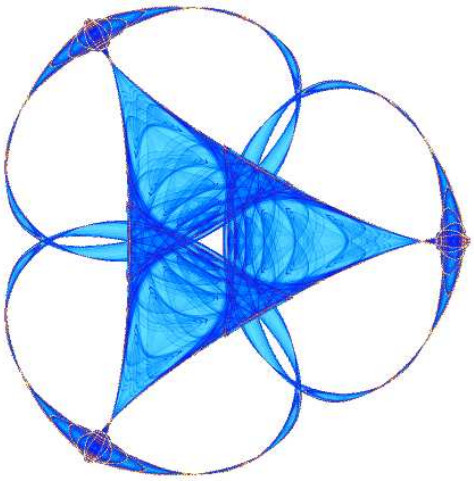
Vicent Caselles

and

Guillermo Sapiro

IMA Preprint Series # 2262

(June 2009)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

400 Lind Hall

207 Church Street S.E.

Minneapolis, Minnesota 55455–0436

Phone: 612-624-6066 Fax: 612-626-7370

URL: http://www.ima.umn.edu

A Unifying Framework for Image Inpainting

Aurélie Bugeau, Marcelo Bertalmío, Vicent Caselles, and Guillermo Sapiro

Abstract

Inpainting is the art of modifying an image in a form that is not detectable by an ordinary observer. There are numerous and very different approaches to tackle the inpainting problem, but we point out that the most successful inpainting algorithms are based on one or two of the following three basic techniques: copy-and-paste texture synthesis, geometric PDE's, and coherence among neighboring pixels. We combine these three building blocks in a unifying variational model, and provide a working algorithm for image inpainting trying to approximate the minimum of the proposed energy functional. Our experiments show that the combination of all three terms of the proposed energy works better than taking each term separately, and the results obtained are state-of-the-art.

Index Terms

Image inpainting, variational models, texture synthesis, PDE's.

I. Introduction

Inpainting is the art of modifying an image or video in a form that is not (easily) detectable by an ordinary observer, and has become a fundamental area of research in image processing. This interest in image inpainting is born in part in the great practical importance of restoring and modifying images and videos, but is also a result of the value of using image inpainting to understand the validity of different image models. Considering image models then, image inpainting techniques can be divided in a number of basic groups. On one group we have the works that consider geometric smoothness and are closely related to the Gestalt principle of *continuation*. The key idea here is that the available image information is smoothly continued into the regions to be filled-in or to be modified, see for example [1], [2], [3], [4], [5], [6] for a few of the recent contributions in this area. In general, these works perform inpainting via geometric partial differential equations (PDEs) and variational formulations. A different image model leading to inpainting considers the self-similarity principle, meaning that an image has a lot of local repetitions of information. Therefore, the regions to be inpainted can be filled with information gathered from other areas of the image where the information is available. The works in [7], [8], [9] are representative efforts in this direction. It is important to note that these formulations are in general global, using information from large regions of the image, sometimes the entire image (and coherently pasted into the hole as for example in [10], see next section), while those formulations based on models of smooth continuation are coherent but only local, primarily propagating information

A. Bugeau is with Barcelona Media - Centre d'Innovaci, Barcelona, Spain, aurelie.bugeau@barcelonamedia.org.

M. Bertalmío is with Departament de Tecnologia, Universitat Pompeu-Fabra, Barcelona, Spain and Barcelona Media - Centre d'Innovaci, Barcelona, Spain, marcelo.bertalmio@upf.edu.

V. Caselles is with Departament de Tecnologia, Universitat Pompeu-Fabra, Barcelona, Spain, vicent.caselles@upf.edu.

G. Sapiro is with Department of Electrical and Computer Engineering University of Minnesota, Minneapolis, MN 55455, USA, guille@ece.umn.edu.

from areas immediately surrounding the regions to be processed. More recently, yet another image model has been successfully used for image inpainting, this based on the consideration that image patches can be efficiently represented as a sparse linear combination of atoms from a learned dictionary. Since such atoms are complete (learned from image datasets), the filling of missing information is automatically achieved, see [11]. This model is particularly powerful for relatively small holes.

These different image models can be simultaneously considered, since they are not necessarily contradictory and none of them can address all types of natural images and all kinds of regions of missing information or regions to be modified. For example, [12] proposed a way to combine PDEs for smooth geometric continuation with self-similarity approaches for texture synthesis. This is done by first decomposing the image into a piecewise smooth component, ideal for the PDEs approach, and a textured repetitive component, ideal for the self-similarity model. The decomposition folows [13]. More recently, combinations between the self-similarity and the sparsity models started to appear as well [14], [15].

It is in this line of research that this paper emerges. In particular, we propose a novel variationally inspired image inpainting model that not only leads to state-of-the-art results but also formally unifies and connects these fundamental image models. Learning from the past best and more effective models, we propose and combine variational terms that are born from texture synthesis (self-similarity), with those that come from geometric approaches (smooth continuations) and those that promote a coherent and smooth filling-in of missing information. A unique characteristic of our proposed framework is that by unifying and combining previous approaches and models, we extend the class of data that can be handled by a unique technique. The proposed formulation works on the patch domain, following the success of numerous recent image processing algorithms.

The remainder of this paper is organized as follows. In Section II we describe the three models which are common to all the most successful inpainting algorithms, and propose to combine them with our unifying variational model. In Section III we discuss in detail the first energy term, corresponding to texture synthesis. In Section IV we discuss the second and third terms, corresponding to geometry and coherence. Section V covers the implementation details of our algorithm, and Section VI presents some experimental results and comparisons with state of the art techniques. Finally, Section VII concludes the paper.

A. Notation

The set of image pixels is denoted as $\mathcal{P}$, and $u : \mathcal{P} \to [0, 255]^{dim}$ represents the image grey valued intensity ($dim = 1$) or color ($dim = 3$) values. Each image contains a region Ω that has to be inpainted/filled-in. The set of all pixels that do not belong to this region is denoted by Ω^c (*i.e.*, $\mathcal{P} = \Omega \cup \Omega^c$). The mask function M indicates if a pixel $p \in \mathcal{P}$ belongs ($M(p) = 0$) or not ($M(p) = 1$) to Ω. We denote by Ψ_s the patch (square set of pixels) of size $|\Psi_s| = (2s + 1) \times (2s + 1)$ centered at 0: $\Psi_s = \{k = (k_1, k_2) \,|\, k_1, k_2 \in (-s, s)\}$. The following abusive notation may be used as well, $\sum_{k \in \Psi_s} \cdot := \sum_{k_1=-s}^{s} \sum_{k_2=-s}^{s} \cdot \quad$. By $P_s(p)$ we denote the $(2s + 1) \times (2s + 1)$ *image patch* centered on p. In practice we will exclude from Ω^c all pixels which are closer than a distance s to the image borders, such that patches centered at $p \in \Omega^c$ are always completely inside the image. Finally, as in [16] we define a *correspondence map* or mapping function $\varphi : \mathcal{P} \to \Omega^c$ as a function that associates to each

image pixel another pixel in Ω^c such that

$$\varphi(p) = \begin{cases} p = (x, y) & \text{if } p = (x, y) \in \Omega^c, \\ q \in \Omega^c & \text{if } p \in \Omega. \end{cases} \tag{1}$$

II. Building Blocks from Prior Art

As mentioned in the introduction, the literature on image inpainting is quite extensive, and reviewing it is out of the scope of this paper. In this section we only concentrate on the building blocks for our proposed framework.

In the seminal paper [8], and motivated by classical Shannon concepts, Efros and Leung presented a simple yet effective non-parametric texture synthesis method based on image patches. The texture is modeled, as in a graphical model, assuming that the probability distribution of brightness values for one pixel given the brightness values of its spatial neighborhood is independent from the rest of the image. The neighborhood is a square window around the pixel and its size is a global (and critical) parameter of the algorithm. The input of the algorithm is a set of model image patches (all patches entirely belonging to Ω^c) and the task is to select an appropriate patch for each of the unknown pixels (pixels in Ω), so as to predict its value. This is done by computing a distance between the known neighborhood of an unknown pixel and each of the input patches. More precisely, considering that the distance d between the pixel p and the pixel q is a sum of squared differences (SSD) defined as

$$d(p, q) := \sum_{k \in \Psi_s} \|u(p + k) - u(q + k)\|^2, \tag{2}$$

finding the candidate pixel $\hat{p}$ for an unknown pixel $p \in \Omega$ corresponds to solving

$$\hat{p} = \arg\min_{q \in \Omega^c} \; d(p, q). \tag{3}$$

In practice, only the known neighborhood of the unknown pixel is taken into account in the distance computation, which leads to computing:

$$\hat{p} = \arg\min_{q \in \Omega^c} \sum_{k \in \Psi_s} M(p + k)\|u(p + k) - u(q + k)\|^2, \tag{4}$$

corresponding to using the following distance measure:

$$d^*(p, q) := \sum_{k \in \Psi_s} M(p + k)\|u(p + k) - u(q + k)\|^2. \tag{5}$$

Algorithm 1 recaps this simple texture synthesis process. The pixels are inpainted in a certain predefined order (see [7] for more on this), for example, from the boundary inwards.

Algorithm 1 Texture synthesis algorithm from [8]

while $\Omega \neq \emptyset$ **do**
1- Select a pixel p on the boundary $\delta\Omega$ of Ω,
2- Find $\hat{p}$ solving Equation (4),
3- Set $u(p) = u(\hat{p})$, $M(p) = 1$, $\Omega = \Omega \backslash p$.
end while

The texture synthesis problem as just described then consists in finding for each pixel in Ω the intensity value minimizing

Equation (4). This problem is similar to finding the *correspondence map* [16], $\varphi : \mathcal{P} \to \Omega^c$, that associates each pixel of the image to an original pixel such that

$$\varphi(p) = \begin{cases} p = (x, y) & \text{if } p = (x, y) \in \Omega^c, \\ \arg\min_{q \in \Omega^c} \; d^*(p, q) & \text{if } p \in \Omega. \end{cases} \tag{6}$$

As just described and considered in [8], this texture synthesis approach is a one-pass greedy algorithm: once a pixel is filled in, its value remains unchanged. This technique then suffers from common problems with greedy algorithms, being the filling order (as well as the patch size) very critical. This is reflected for example in propagating errors of wrongly selected filling pixels (note that filled pixels are used in future iterations as part of the known data, the mask M changes once a pixel is filled), often observed as drifting artifacts. While staying in the greedy one-pass arena, the authors of [7] designed a clever ordering procedure with priorities depending on edge strength, further improving on [8]. Our proposed image inpainting framework includes a (non-greedy) texture synthesis component inspired by these works. Such component, detailed in the next section, is formulated in a variational framework acting on the patch space.

Following this line of research of copying patches to the regions to be filled-in, a number of authors proposed to add spatial (and temporal for video) coherence in the texture synthesis process. The first such work was by Ashikmin [17], who proposed to perform exemplar-based texture synthesis looking for the best match for pixel $p \in \Omega$ not in all Ω^c but only among the set $\{\varphi(p + k) - k\}$ of shifted candidates from the correspondents of the neighbors of p. The idea, as stated by the author, was to increase performance by not "starting the search process from scratch at each new pixel," but in practice it also imposes a certain *coherence* in the mapping function φ which clearly improves the visual quality of the synthesis results. This idea was later formalized a bit more and the name "coherence" introduced in this context in [18]. In [10], both spatial and temporal coherence for video inpainting are imposed via a variational formulation globally optimized for. Coherence appears also in the inpainting formulation of [19] by favoring the similarity of the overlapping region of patches corresponding to neighboring pixels. There are also a number of works (see [11], [20] and references therein) which perform inpainting in a context of sparse representation: patches inside Ω are synthesized as a sparse linear combination of elements from an image dictionary as reduced as possible. The sparsity of the linear combination, the compactness of the dictionary and the use of overlapping patches favor mapping functions φ which are *coherent*. Since coherence at the level of image patches has been found to be critical for state-of-the-art image inpainting, such a term will be explicitly incorporated in the variational formulation here proposed.

As mentioned in the introduction, in parallel to the texture synthesis work, there is a whole body of geometric PDEs and variational formulations that have been very successfully used for inpainting problems, in particular for piecewise smooth images or when the gap Ω is thinner than the surrounding objects. The first work in this area was by Masnou and Morel [5], whose algorithm performs inpainting by joining with geodesic curves the points of the level lines (iso-value contours) arriving at the boundary of Ω. They were inspired by the work of Nitzberg et al. [21] who, in the context of image segmentation, were looking for completion curves in a missing region Ω which should be as short as possible and should respect Gestalt's *principle of good (smooth) continuation*. In order to join with a curve C two points p, q lying on the boundary $\partial\Omega$ of Ω,

Nitzberg et al. proposed to choose the curve minimizing Euler's elastica:

$$\int_C (\alpha + \beta\kappa^2) ds, \tag{7}$$

where the minimum is taken among all curves C (inside Ω) joining p and q (with tangents τ_p and τ_q at p and q, respectively). κ denotes the curvature of C, ds its arc length, and α, β are positive constants. This functional is not lower semicontinuous and its minimization leads to a fourth-order PDE, so Ballester et al. [1] proposed a relaxation of the Elastica to perform image inpainting:

$$\int_\Omega [|\text{div}(\theta)|^p (a + b(|\nabla G * u|)] d\Omega, \tag{8}$$

where $u : R^2 \rightarrow R$ is the image to be inpainted, $\theta : R \rightarrow R^2$ is the normalized gradient (and therefore $div(\theta) = \kappa$), G is a smoothing kernel such as a Gaussian filter, and $p \geq 1$, $a, b > 0$ are parameters. The Euler-Lagrange of this functional leads to a system of two coupled second order PDE's for u and θ. The basic idea is that, as we iteratively run these PDE's, the level lines of the image u are smoothly transferred from a band around Ω toward the inside of Ω, while keeping the image u and its gradient direction θ compatible. Many other functionals for inpainting have been proposed, as well as methods that directly introduce a PDE to perform inpainting, although these PDE's do not minimize any known functional and therefore these methods are not variational, [3], [22], [23], [24]. The underlying idea of these type of approaches is to smoothly transport the geometry of the image into the region being inpainted. These type of algorithms are thereby very efficient at smoothly inpainting contours, but for the same reason fail in textured parts of the image.

The geometric smoothness and transportation concept will be part of our proposed framework as well, via an additional term in the variational formulation. In contrast with the classical approaches where this is addressed at the pixel level, we incorporate it at patch level (there are a number of works that combine geometric PDE's and variational formulations with a setting on patches, e.g. see [25], [26] and references therein). This will make this additional term compatible with the other two terms mentioned above, which are derived from texture synthesis work.

With these techniques in mind, we next introduce our model, that unifies and extends on these concepts.

A. Combining existing approaches into one energy function

As mentioned in the introduction, pairwise combinations of the above basic building blocks are common in the literature (e.g., copy-and-paste texture synthesis with coherence [17], [19], copy-and-paste texture synthesis with geometric PDE's [12], [7], etc). Without intending to build an exhaustive taxonomy of all the inpainting alogorithms that have been proposed in the literature, we have found that all the best of these algorithms are based on one or two, but not all three, of the aforementioned blocks. Our formulation goes one step further and puts together all three models at once.

The energy we propose contains three terms which combine the three main ideas discussed previously. The first term is a variational formulation of the texture synthesis method of Efros and Leung [8], also introduced in [16], [27]. The second and third terms combine both ideas of diffusion and coherence applied in the patch space. We explore two particular diffusion models, the Laplacian isotropic diffusion case and the diffusion inspired in coherence transport [24]. The coherence tries to

reinforce the rigidity of the correspondence map: if pixel x chooses its correspondent $\varphi(x)$, then the pixel chosen by $x+k$ should be near $\varphi(x)+k$.

Thus the proposed energy will have the form

$$\mathcal{E}(\varphi) := \alpha_1 \mathcal{E}_E(\varphi) + \alpha_2 \mathcal{E}_D(\varphi) + \alpha_3 \mathcal{E}_C(\varphi),$$

where φ is the correspondence map, $\alpha_1, \alpha_2, \alpha_3 > 0$ represent weights, $\mathcal{E}_E(\varphi)$ is the energy corresponding to texture synthesis, $\mathcal{E}_D(\varphi)$ is the term corresponding to diffusion (with some coherence), and $\mathcal{E}_C(\varphi)$ reinforces coherence. The precise description of those energies is contained in sections III and IV.

We have found only two works in the literature with related ideas:

-Demanet et al. [16] perform iterative texture synthesis, and suggest as a possible improvement to impose restrictions on the mapping function by penalizing its Total Variation and favoring that it behaves locally as a translation, but the authors did not later pursue this line of thought;

-Aujol et al. [27] proposed and studied several energy functionals for inpainting with terms related to texture synthesis, coherence (by favoring that the mapping function is locally a roto-translation), and geometry (by minimizing the Total Variation of the "geometry" part of the image, after a cartoon+texture decomposition), but the authors did not actually implement a minimization algorithm for them.

Therefore, to the best of our knowledge, the approach introduced in this paper is novel, our contributions being:

-to state explicitly the three fundamental models or building blocks which are common to all the most successful inpainting algorithms;

-to propose an energy functional which synthesizes them;

-to provide a working algorithm for image inpainting trying to approximate the minimum of this energy functional;

-to obtain state-of-the-art results as a consequence of such unifying formulation.

III. First energy term: texture synthesis

A. Iterative texture synthesis

Iterative texture synthesis has already been proposed in many papers [17], [28], [29], [30]. The idea is to use the output image at the previous iteration as initialization for the next iteration. The main motivation for such an approach is texture refinement. In particular, during the initialization (*i.e.*, the first iteration) the inpainting is done from the boundary inwards which often creates edge discontinuities. With the iterative process, these discontinuities as well as the "garbage growing" [8] (propagation of errors) is reduced.

Let us now extend the original texture synthesis algorithm (Algorithm 1) previously presented to an iterative method. The only difference when iterating the process is that, at the end of the first iteration, the mask function M is constant, equal to 1 for all the pixels. From now on, we will refer to this first iteration as the *initialization step* of our method. For the other iterations, we then only need to replace the second step of Algorithm 1 "Find $\hat{p}$ solving Equation (4)" by "Find $\hat{p}$ solving

Equation (3)". Using the correspondence map φ, Equation (2) becomes

$$d_1(p,q,\varphi) = \sum_{k\in\Psi_s} \|u(\varphi(p+k)) - u(q+k)\|^2 \tag{9}$$

and the iterative algorithm can be defined (Algorithm 2).

Algorithm 2 Iterative texture synthesis algorithm

A- INITIALIZATION STEP:
1- Set $\Omega^0 = \Omega$
2- *Process all the pixels of the mask:*
while $\Omega^0 \neq \emptyset$ **do**
a- Select a pixel p on the boundary $\delta\Omega^0$ of Ω^0,
b- Find $\varphi^0(p)$ solving

$$\varphi^0(p) = \arg\min_{q\in\Omega^c} d^*(p,q), \tag{10}$$

c- Set $u(p) = u(\varphi^0(p))$, $M(p) = 1$, $\Omega^0 = \Omega^0 \backslash p$.
end while

B- ITERATIVE PROCESS:
- Set $n = 1$
repeat
1- Set $\Omega^n = \Omega$
2- *Process all the pixels of the mask:*
while $\Omega^n \neq \emptyset$ **do**
a- Select a pixel p on the boundary $\delta\Omega^n$ of Ω^n,
b- Find $\varphi^n(p)$ solving

$$\varphi^n(p) = \arg\min_{q\in\Omega^c} d_1(p,q,\varphi^{n-1}), \tag{11}$$

c- Set $u(p) = u(\varphi^n(p))$, $\Omega^n = \Omega^n \backslash p$.
end while
3- Set $n \leftarrow n+1$
until Convergence

In the algorithm, the superscript n indicates the index of the iteration. Note that the function M does not appear anymore in Equation (11). Indeed, all the pixels have already been inpainted and we use their color at iteration $n-1$ to compute the mapping function (and hence the new color) at iteration n.

B. Connection with energy minimization

The texture synthesis problem can be recasted as an energy minimization problem. Indeed, by applying the texture synthesis algorithm (Algorithm 1), one intends to minimize the following energy function:

$$E_0(\varphi) = \sum_{p\in\Omega} \sum_{k\in\Psi_s^*(p)} \|u(p+k) - u(\varphi(p)+k)\|^2, \tag{12}$$

where Ψ_s^* denotes the patch containing only the known neighbors of p. Nevertheless, by minimizing a sum of squared differences on each pixel and applying the greedy Algorithm 1, we can not ensure the global minimization of the energy.

In order to clarify the difference between minimizing the previous energy and applying Algorithm 1, let us give a simple example. Suppose that we are trying to inpaint an image containing repetitive patterns. An example of such an image is shown

on Figure 1a) and the results obtained after applying Algorithm 1 with two different patch sizes are shown on figures 1b) and 1c). As it can be noticed the result is perfect with 9×9 patches but not with 5×5 patches. Since this image contains repetitive patterns, we know that the global minimum of the Energy (12) is equal to 0 because the gap Ω can be filled by verbatim copy of a piece of Ω^c, with the same shape and size as Ω. If we compute the value of the energy (12) after Algorithm 1 has been applied, we see that for 9×9 patches it is indeed equal to 0, while for 5×5 patches it returns a much larger value.

a) Original image b) 9×9 patches c) 5×5 patches

Fig. 1. Result of algorithm 1 for an image containing repetitive patterns. The mask to be inpainted is the gray rectangle.

The previous result highlights two interesting properties: firstly, the size of the patches has to be chosen carefully; secondly, even in the presence of repetitive patterns the minimum of Energy (12) may not always be reached using a non iterative texture synthesis process. A solution to refine the inpainting and get closer to the minimum is to use the iterative approach (Algorithm 2).

By using the iterative algorithm, the energy that we would like to minimize can be written as:

$$\mathcal{E}_E(\varphi) = \sum_{p \in \Omega} \sum_{k \in \Psi_s} \|u(\varphi(p+k)) - u(\varphi(p)+k)\|^2, \tag{13}$$

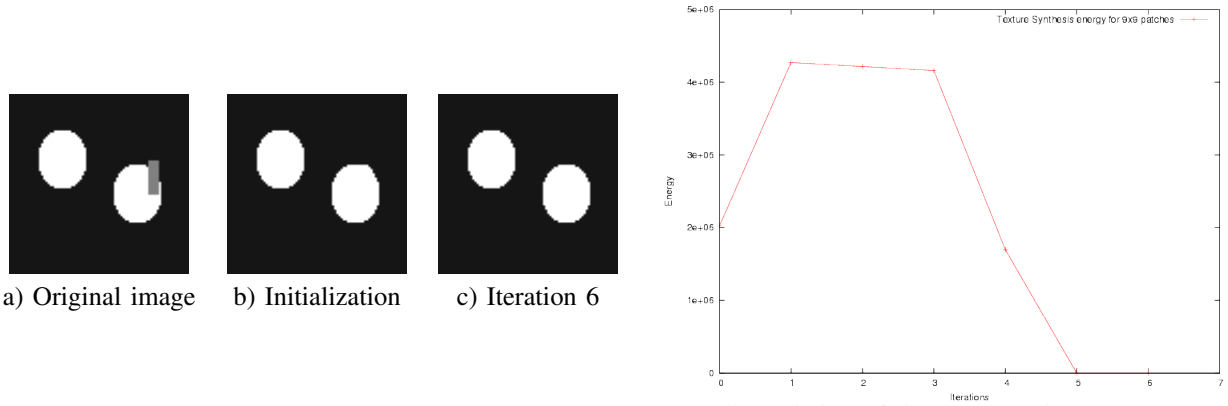
or, equivalently, as:

$$\mathcal{E}_E(\varphi) = \sum_{p \in \Omega} d_1(p, \varphi(p), \varphi). \tag{14}$$

Notice that this is a highly non linear and non convex energy for the correspondence map whose Euler-Lagrange equations are too complex to be addressed directly due to the high number of dimensions of the unkonwns. For that reason we adopted the practical point of view of fixing the map $\varphi = \varphi^{n-1}$ on $u(\varphi(p+k))$ and minimizing with respect to it on the other argument. This seems to be a common approach [16].

Let us show the influence of the iterative algorithm on the value of energy (14) for the repetitive image pattern. On the example of Figure 1, we saw that the minimum of the energy (equal to 0) was not reached with a single-pass algorithm with patch size 5×5. On Figure 2 we show how this is solved with the iterative algorithm, which refines the inpainting and may allow converging to the global minimum. On Figure 2d), it might seem surprising that the energy increases in the first step but remember that for the initialization we have to use the mask M, which does not appear in energy (14).

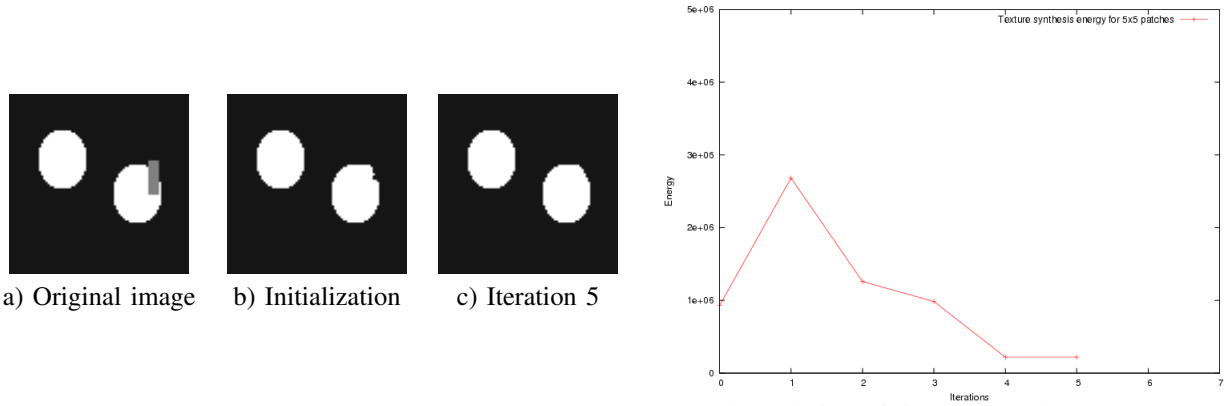
Even if the iterative process can give a better approximation of the minimum of the energy functional defined by Equation (14), we still can not guarantee formally that we are minimizing it globally. The results on figure 3 show that this iterative approach can still lead to falling into a local minimum, because of the patch size. Furthermore, we have, until now, only been talking about images containing perfect repetitive patterns, for which we know that the minimum of the energy must be equal to zero (if the mask does not cover all the repetitions). For other images, it is hard to give an analysis of the algorithm since



a) Original image b) Initialization c) Iteration 6

d) Evolution of the energy value, eq. (14)

Fig. 2. Result of iterative texture synthesis algorithm for an image containing repetitive patterns. The mask to be inpainted is the gray rectangle and the size of the patches is 9×9.



a) Original image b) Initialization c) Iteration 5

d) Evolution of the energy value, eq. (14)

Fig. 3. Result of iterative texture synthesis algorithm for an image containing repetitive patterns. The mask to be inpainted is the gray rectangle and the size of the patches is 5×5. This result shows the convergence to a local minimum.

we do not know which is the minimum of the energy and therefore we can't tell whether the algorithm converges to a local or a global minimum, but we have experimentally confirmed that the energy does indeed decrease, often uniformly after the first iteration.

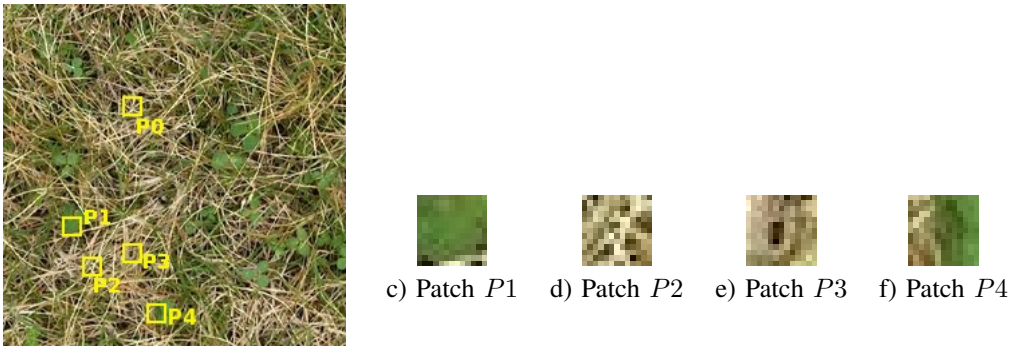
C. Choosing the distance function

As explained in the previous section, in order to minimize the first term of the proposed energy we may assign to each pixel $p \in \Omega$ a correspondent pixel $q \in \Omega^c$, $\varphi(p) = q$, such that the distance d_1 between p and q is minimal. The distance function d_1 (9) is a simple Sum of Squared Differences (SSD) and, surprisingly, it is the distance function used in all patch-based texture synthesis algorithms. We have found that the SSD may be problematic for a wide class of images, especially when we have to iterate in order to obtain the solution, as it is in our case. Figure 4 shows that, as early as at the second step of the iterative process, the bias of the SSD for uniform patches makes a flat region take over most of the inpainting region Ω. The figure also shows that while at the first iteration the mapping function φ is rather "dispersive," already at the second iteration it starts to behave locally as a translation toward flat regions. Figure 5 explains what is the problem with the SSD. We have a

textured patch such as $P0$ which is, *in its overall appearance*, very similar to $P2$. But, pixelwise, as the SSD is computed, the difference between pixels at the same location in both patches is greater than the difference between the pixels in $P0$ and the *average value* of $P0$. Think of the difference between two sinusoids of average μ which are 180^{deg} out of phase: although they look exactly the same, their SSD is greater than the SSD between either sinusoid and the constant function of value μ. This is why patch $P4$, which is rather uniform and does not resemble at all $P0$, is more similar *in terms of SSD* to $P0$ than $P2$ is to $P0$. Based on these observations, we should look for another distance measure or, conversely, impose certain restrictions on the mapping function φ that prevents it to become locally a translation towards uniform patches. There are several works in the literature which introduce some assumptions on φ: in [16] the authors propose to encourage φ to locally look like the identity function, in [27] the solutions are searched within a set of piecewise roto-translations. As we mention in the conclusion, the introduction of restrictions on φ to maintain its "dispersive" nature is subject of further research; in this paper we have opted to define a new distance measure which is more suitable than the SSD for iterative texture synthesis.

a) Original image b) Initialization c) Iteration 2 d) φ at initialization e) φ at iteration 2

Fig. 4. Result after two iterations of the iterative algorithm on a grass image with 9×9 patches. The last two images display the value of the correspondence map for several pixels of the mask.



a) Original image

b) Patch $P0$ c) Patch $P1$ d) Patch $P2$ e) Patch $P3$ f) Patch $P4$

Fig. 5. Sum of squared differences between texture and smooth patches: $d_1(P0, P1) = 5290$; $d_1(P0, P2) = 5443$; $d_1(P0, P3) = 5416$; $d_1(P0, P4) = 5070$.

1) Modifying the distance measure: By observing the five patches on Figure 5, it can be noticed that a smooth patch can be differentiated from a textured patch thanks to its distribution. In order to compute the distance between two distributions ρ_0 and ρ_1 corresponding to two patches P_p and P_q, we use the Bhattacharyya probability density distance:

$$d_B(P_p, P_q) = \sqrt{1 - \sqrt{\sum_{i=1}^{B} \rho_p(i)\rho_q(i)}}, \tag{15}$$

where ρ_p is the histogram of the patch centered on p, ρ_q is the histogram of the patch centered on q, B is the number of bins when the distributions are discrete and defined as histograms. In practice a histogram is computed on each patch by using 8 bins for each color dimension (*i.e.* 512 bins in the case of color images). For the patches of Figure 5, we get the following values:

$$d_B(P0, P1) = 0.96 \ ; \ d_B(P0, P2) = 0.47 \ ;$$

$$d_B(P0, P3) = 0.46 \ ; \ d_B(P0, P4) = 0.86 \ .$$

As this measure is invariant to rotations, it is still not adequate to perform texture synthesis. That is why we chose to combine it with the SSD and propose the following *texture distance*:

$$d_T(P_p, P_q) = \frac{1}{|\Psi_s|} d(P_p, P_q) d_B(P_p, P_q), \tag{16}$$

which, for the cases illustrated in Figure 5, approximately leads to the following values:

$$d_T(P0, P1) = 5081 \ ; \ d_T(P0, P2) = 2557 \ ;$$

$$d_T(P0, P3) = 2493 \ ; \ d_T(P0, P4) = 4361 \ .$$

Of course when introducing this distance into the iterative texture synthesis algorithm, the histogram ρ_p of the patch centered on p has to be updated at each iteration and therefore depends on φ^{n-1}. The algorithm obtained will be named *texture distance iterative algorithm,* and corresponds to introducing the texture distance into equations (10) and (11). A result of this algorithm is shown on Figure 6. Note how the blur has disappeared with respect to Figure 4c).

2) Pruning: Even if the results obtained by changing the distance are improved, they lead to one major drawback: the computational cost. Indeed computing a histogram for each patch is extremely time-consuming. Recently, an algorithm to speed-up patch searches has been proposed in [31], using randomized sampling and exploiting the coherence in natural images. In order to get a faster algorithm we propose to use a method that we name *pruning*, inspired by the label pruning of [19]. We reduce the number of possible candidates for each pixel of Ω after the initialization step has been performed. The number N of candidates kept for each pixel is a fixed parameter set by the user. These candidates are simply the ones that had the smallest texture distance to the pixel during the initialization step. In the sequel, we will denote as $C(p)$ the *pruning set* (with N candidates) for the pixel p. While the initialization still has an important computational cost, the subsequent iterations are now much faster, and the visual quality does not deteriorate when N is high enough. A result of this approach on the grass image is presented in Figure 6, second row.

Despite the pruning, the processing of each iteration still requires quite some time (even without taking into account the initialization step), in particular due to the fact that a histogram still has to be computed for each patch. We have therefore run some experiments where we apply the texture distance d_T only at the initialization, use it to perform the pruning, and then in

a) Initialization b) Iteration 2 c) φ at initialization d) φ at iteration 2

Fig. 6. Result after two iterations of the texture distance iterative algorithm on the grass image with 9×9 patches. First row: using d_T for all iterations. Second row: with pruning and using d_T for all iterations. Third row: with pruning, using d_T for the first iteration, SSD for all the others.

all subsequent iterations use just the SSD to compare patches. The results are very similar to those obtained when using d_T at every iteration, as we can see in the bottom row of Figure 6, but at a much lower computational cost. This makes sense since d_T has been introduced in order to reduce the appearance of flat regions in the results, and by applying d_T for the pruning we are discarding from the pruning set all pixels that could produce said flatness. We call this improved algorithm (Algorithm 3) the *texture distance and pruning iterative algorithm*.

IV. Second and third energy terms: Geometry and Coherence

A. The structure of the algorithm and the initialization

Before going any further and defining the diffusion and coherence energies, let us explain the general structure of the algorithm. The energy has the form

$$\mathcal{E}(\varphi) = \sum_{p \in \Omega} D(\varphi(p), \varphi|_{\Psi_s \setminus \{p\}}), \tag{19}$$

where $D(\varphi(p), \varphi|_{\Psi_s \setminus \{p\}}) := \sum_{i=1,2,3} \alpha_i D_i(\varphi(p), \varphi|_{\Psi_s \setminus \{p\}})$ is the energy density (sum of the texture synthesis, diffusion, and coherence terms) which depends on $\varphi(p)$ and on φ restricted to the rest of the patch centered at p ($\varphi|_{\Psi_s \setminus \{p\}}$), and $\alpha_i > 0$ are weights.

Due to the difficulties in minimizing in practice such a general form of the energy, we propose to proceed in an iterative

Algorithm 3 Texture distance and pruning iterative algorithm

A- INITIALIZATION STEP:
1- Set $\Omega^0 = \Omega$
2- *Process all the pixels of the mask:*
while $\Omega^0 \neq \emptyset$ **do**
a- Select a pixel p on the boundary $\delta\Omega^0$ of Ω^0,
b- Find $\varphi^0(p)$ solving

$$\varphi^0(p) = \arg\min_{q \in \Omega^c} d_T(p, q), \tag{17}$$

c- Fill in the pruning set $C(p)$ with the N best candidates.
d- Set $u(p) = u(\varphi^0(p))$, $M(p) = 1$, $\Omega^0 = \Omega^0 \backslash p$.
end while

B- ITERATIVE PROCESS:
- Set $n = 1$
repeat
1- Set $\Omega^n = \Omega$
2- *Process all the pixels of the mask:*
while $\Omega^n \neq \emptyset$ **do**
a- Select a pixel p on the boundary $\delta\Omega^n$ of Ω^n,
b- Find $\varphi^n(p)$ solving

$$\varphi^n(p) = \arg\min_{q \in C(p)} d_1(p, q, \varphi^{n-1}), \tag{18}$$

c- Set $u(p) = u(\varphi^n(p))$, $\Omega^n = \Omega^n \backslash p$.
end while
3- Set $n \leftarrow n + 1$
until Convergence

way, minimizing at each iteration the restricted energy

$$\mathcal{E}(\varphi, \varphi^{n-1}) = \sum_{p \in \Omega} D(\varphi(p), \varphi^{n-1}|_{\Psi_s \backslash \{p\}}), \tag{20}$$

where φ^{n-1} is the current estimate of φ.

Then we may use that

$$\min_{\varphi} \mathcal{E}(\varphi, \varphi^{n-1}) = \sum_{p \in \Omega} \min_{\varphi} D(\varphi(p), \varphi^{n-1}|_{\Psi_s \backslash \{p\}}), \tag{21}$$

which permits to devise an effective search algorithm (the one we use in practice) although this may not correspond to the actual global minimum of (19), because of the boundary conditions do not necessarily satisfy. This can be easily seen by considering the functional $F(\varphi) = \sum_{i=1}^{N} |\varphi(i) - \varphi(i-1)|^2$ with the boundary conditions $\varphi(0) = 0$ and $\varphi(N) = N$. The minimum of F is $\varphi(i) = i$. If we replace F by $F(\varphi, \varphi^{n-1}) = \sum_{i=1}^{N} |\varphi(i) - \varphi^{n-1}(i-1)|^2$ and we take $\varphi^{n-1}(i) = i$ (the actual minimum of F,) then $\min_{\varphi} F(\varphi, \varphi^{n-1})$ would produce $\varphi(i) = i - 1$ for any $i = 1, \ldots, N-1$. In spite of this, we shall adopt this practical point of view, since the first energy term already ensures that the boundary conditions are met.

With this structure, the algorithm requires an initialization. This step is different from the other iterations as the whole image is not yet known. In the previous section we have proposed to use a different distance for this iteration.

As we will explain in this section, both for the geometry and the coherence terms it is easier to compute the distances when the initialization step has already been performed and therefore the whole image is known. The initialization involves only the texture synthesis term, with d_T, while the iterative process involves the three terms with the SSD distance over the pruning

set for each pixel in Ω.

B. The geometry term

Let us denote $u^\varphi(p) = u(\varphi(p))$, $p \in \Omega$. If we choose Laplacian diffusion for the geometry term, then it could be written as

$$\mathcal{E}_D(\varphi) = \sum_{p\in\Omega} \sum_{k\in\Psi_s} |\Delta u^\varphi(p+k)|^2,$$

where $\Delta u^\varphi(p)$ denotes the discrete Laplacian of u^φ at p. This is essentially equivalent to a multiple of $\sum_{p\in\Omega} |\Delta u^\varphi(p)|^2$.

Given a pixel $p = (x, y)$, let us denote its four neighboring pixels by $p^E = (x+1, y)$, $p^W = (x-1, y)$, $p^S = (x, y+1)$ and $p^N = (x, y-1)$. Then the discrete Laplacian operator is

$$\Delta u^\varphi(p) = u^\varphi(p^W) + u^\varphi(p^E) + u^\varphi(p^S) + u^\varphi(p^N) - 4u^\varphi(p). \tag{22}$$

As we did in Section III, here we use an iterative algorithm updating the correspondence map step by step. For that, assume that we already know φ^{n-1} and we intend to compute φ^n. Let us denote $\psi = \varphi^{n-1}$, and let

$$v(p, \psi) = \frac{1}{4}[u^\psi(p^W) + u^\psi(p^E) + u^\psi(p^S) + u^\psi(p^N)]. \tag{23}$$

Then we may write the (iterative) geometry term $\mathcal{E}_D$ as

$$\sum_{p\in\Omega} \sum_{k\in\Psi_s} |v(p+k, \psi) - u^\varphi(p+k))|\, dp. \tag{24}$$

In order to reinforce the coherence we would like to impose that

$$u(\varphi(p+k)) \approx u(\varphi(p) + k), \tag{25}$$

and we redefine (24) as

$$\mathcal{E}_D(\varphi, \varphi^{n-1}) = \sum_{p\in\Omega} d_2(p, \varphi(p), \varphi^{n-1}). \tag{26}$$

where

$$d_2(p, q, \psi) = \sum_{k\in\Psi_s} |v(p+k, \psi) - u(q+k)|. \tag{27}$$

Minimizing eq. (26) amounts then to finding the correspondence map that best matches the image u with the image v defined in eq. (23). We call v the *diffusion image*. In the case just mentioned, with Laplacian diffusion, the definition of v comes from the definition of the energy. But we may choose instead to specify the diffusion image directly, so that the image information is transported in the direction of the level lines of the image, e.g.

$$v(p, \varphi) = \frac{\sum_{l\in\Psi_r, l\neq 0} \omega(p, l) u(\varphi(p+l))}{\sum_{l\in\Psi_r, l\neq 0} \omega(p, l)}, \tag{28}$$

where $\omega(p, l)$ are the weights proposed in the coherence transport diffusion method of Bornemann [24] ($\omega(p, l)$ is higher when the pixel $(p + l)$ is in the direction of the level line at p).

C. Reinforcing the coherence term

As we already mentioned in Section IV-B, coherence is represented by the relation (25). In this formulation it is implicit that the correspondence map has to be rigid in some sense, it has to assign nearby corresponding points to nearby pixels. We could ask even more and consider that for any $k \in \Psi_s$, the family of values $\{u(\varphi(p+l)+k-l) : l \in \Psi_s\}$ are representative of the value of $u(\varphi(p)+k)\}$ and permit to give an estimate of it. To make this estimate in a robust way we propose to use median instead of mean averages and compute:

$$w(p,k,\varphi) = median_{l\in\Psi_s}\{u\big(\varphi(p+l)-l+k\big)\}. \tag{29}$$

Again, we proceed in an iterative way and we assume that we have φ^{n-1}. Then we compute the value $w(p,k,\varphi^{n-1})$ and define the reinforcement coherence term as

$$\mathcal{E}_C(\varphi,\varphi^{n-1}) = \sum_{p\in\Omega} d_3(p,\varphi(p),\varphi^{n-1}). \tag{30}$$

where

$$d_3(p,q,\psi) = \sum_{k\in\Psi_s} \left[w(p,k,\psi) - u(q+k)\right]^2, \tag{31}$$

is the coherence distance. In other words, the candidate $\varphi(p)$ is chosen so that the intensity of its neighborhood is close to the intensity of the "shifted" candidates of the neighbors of p. In consonance with the previous section we call w the *coherence image*.

D. Combining the three terms

We have now defined three different distance measures, one for texture synthesis (Equation (9)), one for diffusion (Equation (27)) and one for coherence (Equation (31)). These measures are combined into the energy

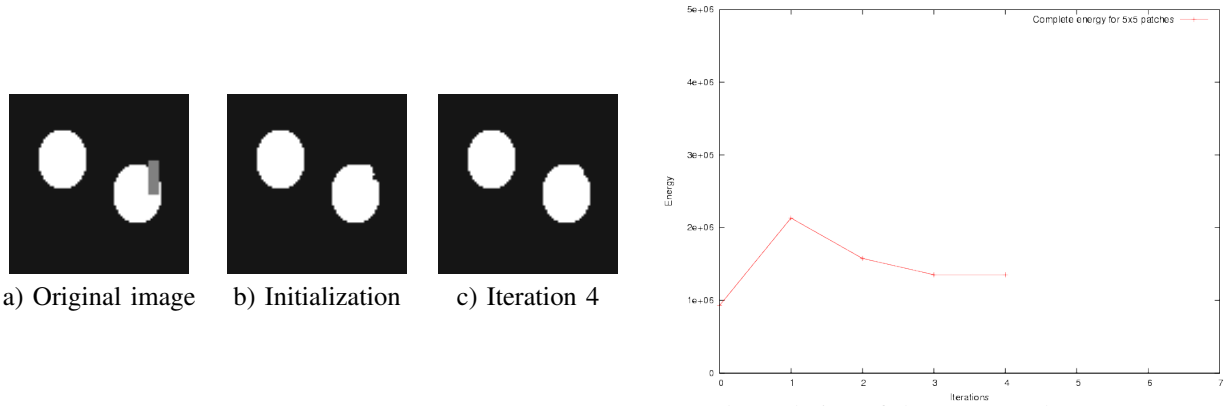
$$\mathcal{E}(\varphi,\varphi^{n-1}) = \sum_{p\in\Omega}\sum_{i=1}^{3} \alpha_i d_i(p,\varphi(p),\varphi^{n-1}), \tag{32}$$

where $\alpha_i > 0$ are some weights. Taking (21) into account we can find the best candidate matching pixel as

$$\varphi(p) = \arg\min_{q\in\Omega^c} \sum_{i=1}^{3} \alpha_i d_i(p,q,\varphi^{n-1}). \tag{33}$$

This new candidate function can directly be incorporated into the texture distance and pruning iterative algorithm (Algorithm 3) by replacing the step B-2-b.

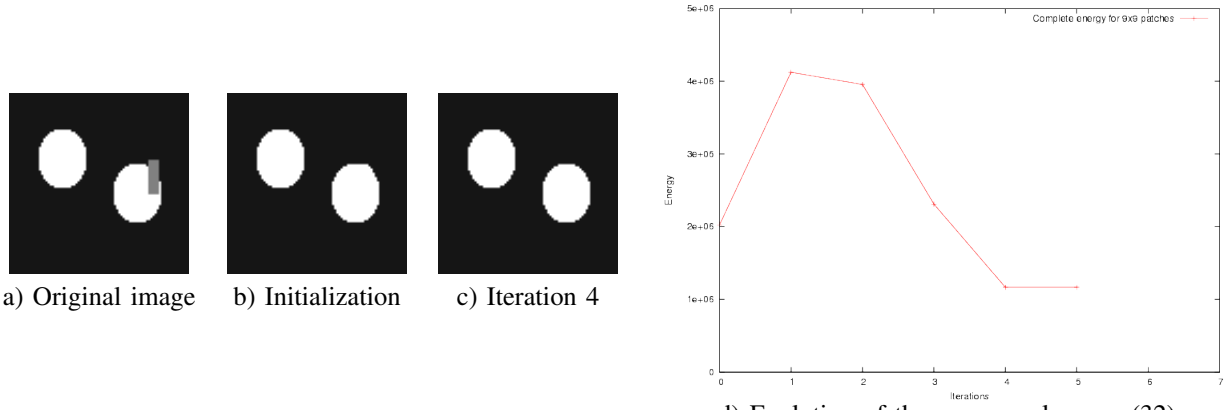
In Section V we shall describe an improved choice of the weights α_i, but for the time being we just choose $\alpha_i = \frac{1}{3}, i = 1,2,3$. We can return to the toy example of Fig. 1 and compute the evolution of the full, three-termed energy value of eq. (32). The results are in Fig. 7 and Fig. 8. As we can see, with 5x5 patches the evolution gets stuck in a local minimum, while with 9x9 patches the image is inpainted correctly. Notice how now the minimum value of the energy associated with the correct inpainting is not zero anymore, something which can be expected because of the presence of the geometry term.



a) Original image b) Initialization c) Iteration 4

d) Evolution of the energy value, eq. (32)

Fig. 7. Result of proposed algorithm for an image containing repetitive patterns. The mask to be inpainted is the gray rectangle and the size of the patches is 5×5.



a) Original image b) Initialization c) Iteration 4

d) Evolution of the energy value, eq. (32)

Fig. 8. Result of proposed algorithm for an image containing repetitive patterns. The mask to be inpainted is the gray rectangle and the size of the patches is 9×9.

V. Implementation details

A. *Multiresolution scheme*

There are several reasons for introducing a multiresolution scheme in our algorithm. Firstly, it permits to accelerate the initialization step, as it is applied on a smaller image. For all other iterations and scales, the full algorithm (with three terms) is applied, with the computational cost savings given by the pruning technique. Secondly, it allows to capture different details of the image, something which could only be achieved without multiresolution by locally changing the size of the patches: if we use the same size for all the scales of the pyramid, different patterns of the image are captured. The larger structures can be captured correctly at the coarsest scales.

Multiresolution schemes have already been used for image inpainting. In particular, in [30], they have been used for enlarging the patches (different types of structural components are captured at different scales) and to guide the search of patches with the already computed neighbors of each pixel. Here we use a slightly different method as the size of the patches is kept the same at all scales and no constraint between scales is imposed (we do not keep or use the values of the inpainted pixels at the previous scale except when initializing the next level). Hence, the algorithm works as follows. First, a Gaussian pyramid is

built from the original image. Next, the iterative inpainting algorithm is applied on the smallest image (the highest one in the pyramid). The result obtained on this image is then used as the initialization for the second level of the pyramid, and so on.

B. Automatic computation of the weights

The weights α_i define the influence of each of the three terms on the result. They have to be defined depending on the properties of the image, and even of each pixel of the image. For example, for a pixel situated in a textured area, the parameter α_1 should be larger than the two other ones, while if it is located on a sharp contour, α_2 should be the largest one. It is more difficult to interpret the influence of α_3 and its value will be given by a similar formula than the ones used for α_1 and α_2.

Before going any further, we first define, for each pixel p, the texture patch $P_1(p)$ extracted from the image at its current iteration as the $(2s+1)\times(2s+1)$ image patch centered at p. The structure patch $P_2(p)$ extracted from the diffusion image and the coherence patch $P_3(p)$ from the coherence image are defined in the same way. In order to define the weights, we assume that their values should depend on the "validity" of the best candidate patch. That is, if no good candidate patch is present in the image for the patch $P_{i,\ i=1\ldots3}$, the value of the corresponding weight should be small. An obvious way to define the validity of a candidate patch $\Psi(q)$, centered on q, is to make it depend on its distance to $P_i(p)$. Therefore the validity of the best candidate patch for the patch $P_i(p)$ depends on the minimal distance:

$$m_i(p) = \min_{q\in\Omega^c} d_i(p,q,\varphi). \tag{34}$$

More specifically, the values of the weights must be inversely proportional to the values of m_i. Because of all the considerations above, we proposed to automatically compute the weights with these functions:

$$\alpha_i(p) = e^{-\frac{m_i}{\sigma}},\ i=1,2,3, \tag{35}$$

where σ is set as

$$\sigma = \frac{m_1+m_2+m_3}{3}. \tag{36}$$

C. Poisson editing

After applying the whole process on several images, we have observed that it may create some illumination discontinuities at the border of the mask. To deal with this defect, we add a post-processing step of Poisson image editing [32]. A small modification is applied to the standard method: instead of taking into account the gradients of the image that has been inpainted, we consider the gradients of the corresponding pixels. That is, for each $p\in\Omega$ we compute $\nabla u(\varphi(p))$ instead of $\nabla u(p)$. This avoids blurring at the boundary of the mask.

VI. Experimental Results

In this section we will test our algorithm regarding variables such as patch size or diffusion method used, and compare it with state-of-the-art inpainting methods.

Fig. 9 compares the results obtained when we consider the three terms of the energy (the proposed algorithm) and when we only take into account one of the terms. Notice how the text in the middle lantern at the bottom row is only successfully inpainted when the three terms are combined.

Fig. 10 compares the results obtained when we use different diffusion images v, either Laplacian diffusion with eq. (23) or coherence transport diffusion with eq. (28): we can see that the differences are minimal, therefore the particular choice of geometric diffusion method does not seem to affect the result.

Fig. 11 compares the results obtained when we use different patch sizes. With small patches the algorithm is unable to inpaint correctly the middle-sized blob at the left of the image. But with big patches another problem arises: the mask Ω is spread all over the image, so there are not enough big patches completely inside Ω^c and the final result is poor. We should incorporate into our algorithm the possibility that the candidate patches are not completely included in Ω^c, and how to do this properly is the subject of further research.

Fig. 12 compares our algorithm with the results obtained with the state of the art techniques of Fadili et al. [20] and Tschumperle et al. [22], as taken from these papers. The differences are most noticeable at the beak, the feathers in the forehead, and the vertical strip to the side of the eye.

On Figure 13, some additional comparisons with existing inpainting methods are presented. We have compared with five other methods already mentioned in the paper: the diffusion methods of [33] and [24], the texture synthesis algorithm from [7], the texture and coherence method from [19], and finally the earlier method we proposed in [34]. All of these methods require the tuning of some parameters. For the method from [33] and [34], as suggested in the original papers, we set the contour preservation parameter p_1 to 0.001, the structure anisotropy $p_2 = 100$, the time step $dt = 150$, and the number of iterations nb equal to 100. For the technique from [24], we set the averaging radius $\epsilon = 5$, the sharpness parameter $\kappa = 25$, the scale parameter for pre-smoothing $\sigma = 1.4$, and for post-smoothing $\rho = 4$. The last parameter that has to be set is the patch size required for both our method and the ones from [7] and [34]. The result of our algorithm shown on Figure 13 has been obtained using Laplacian diffusion, pruning sets of $N = 200$ elements, 20 iterations, three scales ($L = 3$), and 9×9 patches. We can see that our method is able to correctly inpaint the covered lanterns, the text on these lanterns and the red and white striped bars, while the other techniqes fail in some of these aspects.

Figure 14 (c) shows a 800×600 image inpainted with 11×11 patches and four scales. The mask Ω (not shown) covers the kiosk in the center of the image. The reconstruction of the building facade is acceptable (though not perfect,) but the inpainting of the road part of the image has failed. The reason is that this part is not present in the rest of the image, so there are not enough patches to copy from. This is limitation is shared by all methods based on patches. For comparison we show in figure 14 (b) the result obtained by Hays et al. [35], where they search a database of million of images, but in this case the result is not good either.

Figure 15 shows an application of our proposed method to image editing. Here the mask Ω corresponds to a region that we want to relocate in the image, so first we inpaint Ω and then we superimpose the original region at the desired location. We compare with the state of the art techniques of Cho et al. [36] and Barnes et al. [31].

Figure 16 shows an application of our proposed method to image resizing, where we have enlarged the original 500×375

image by adding 100 columns to the right.

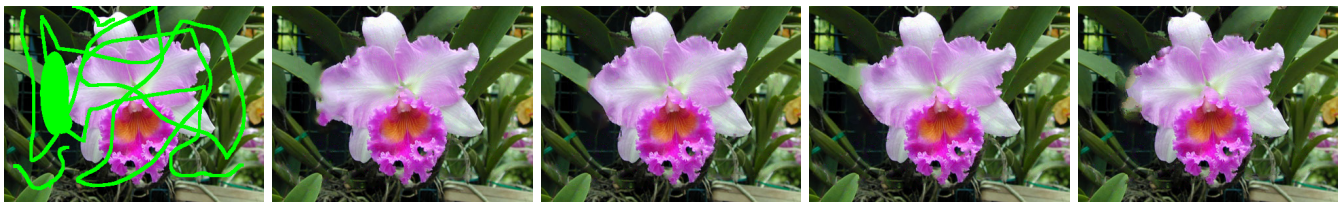
Finally, Fig. 17 shows an image where none of the methods discussed above seems to work. The problem here is that diffusion methods (Fig. 17 (c) and (d)) are able to synthesize new information but they are not able to deal with texture, while patch-based methods (Fig. 17 (e) to (h)) treat texture correctly but are not good at creating new information.

(a) (b) (c) (d) (e)

Fig. 9. Influence of the three terms. (a) Original image. (b) Result of our algorithm. (c) Result using only the texture synthesis term ($\alpha_1 = 1$, $\alpha_2 = \alpha_3 = 0$). (d) Result using only the diffusion term ($\alpha_2 = 1$, $\alpha_1 = \alpha_3 = 0$) . (e) Result using only the coherence term ($\alpha_3 = 1$, $\alpha_1 = \alpha_2 = 0$).

(a) (b) (c)

Fig. 10. Effect of the choice of geometric term. (a) Original image, from [35]. (b) Result with Laplacian diffusion, Eq. (23). (c) Result with coherence transport diffusion, Eq. (28). The differences are minimal, therefore the particular choice of geometric diffusion method does not seem to affect the result.



(a) (b) (c) (d) (e)

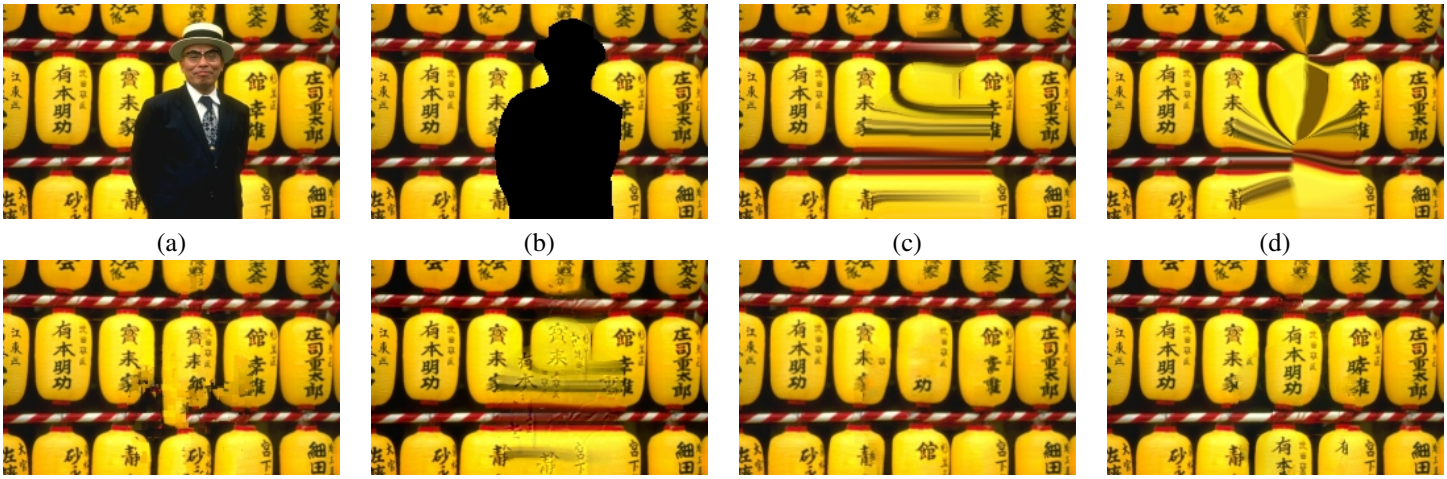
Fig. 11. Effect of the choice of patch size. (a) Original image. (b) Result with 3x3 patches. (c) Result with 5x5 patches. (d) Result with 7x7 patches. (e) Result with 11x11 patches.

VII. Conclusion

In this article we have started by stating explicitly the three models or building blocks which are common to all the most successful inpainting algorithms, which we then combine in a unifying variational model. We also provided a working algorithm for image inpainting trying to approximate the minimum of this energy functional. Our experiments show that the combination of all three terms of the proposed energy works better than taking each term separately, something which seems to be corroborated when comparing our results with other techniques, none of which is based on all the three, but at most two, of the stated building blocks. The particular choice of geometry/diffusion method does not affect noticeably the results, but the

(a) (b) (c) (d) (e)

Fig. 12. Comparisons with other algorithms. (a) Original image. (b) Mask Ω in white. (c) Result from Fadili et al. [20]. (d) Result from Tschumperle et al. [22]. (e) Result from our algorithm. The differences are most noticeable at the tip of the beak, the feathers on the forehead and on the vertical strip to the side of the eye.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 13. Comparison of several inpainting algorithms. (a) Original image. (b) In black the mask to be inpainted. (c) Result from Tschumperle [33]. (d) Result from Bornemann [24]. (e) Result from Criminisi et al. [7]. (f) Result from Bugeau et al. [34]. (g) Result from Komodakis et al. [19] - The result for this last method has been taken directly from the paper, while we have implemented the other techniques. (h) Result obtained with our algorithm.

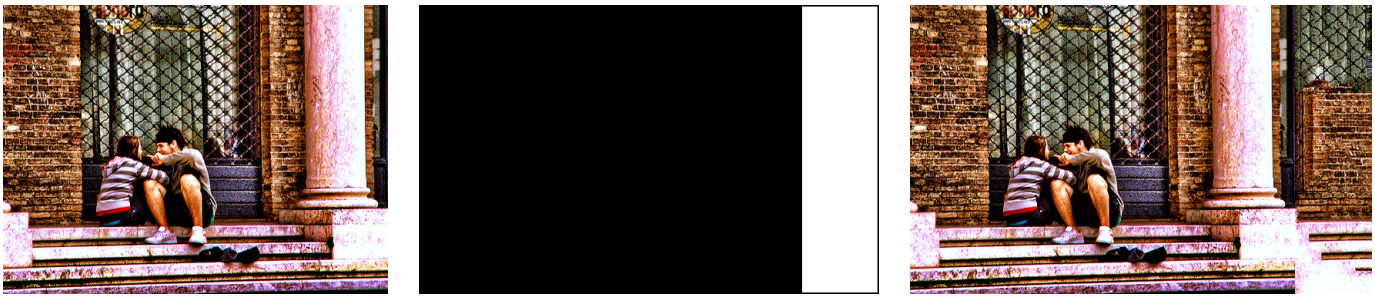
(a) (b) (c)

Fig. 14. Comparison with method by Hays et al. [35]. (a) Original image. (b) Result from [35]. (c) Result from our algorithm.

size of the patches used is a relevant parameter. When the image has not enough patches to copy from, either because the mask is too spread and the patch size is large, or because the mask is placed on a *singular* location on the image (with surroundings that can't be found anywhere else in the image) then the results are poor; this problem is common to all patch-based inpainting methods, and although the presence of a geometry term seems to help it is clearly not enough. We are currently trying to deal with these problems, as well as working on the speeding-up of our algorithm by optimizing the searches in the patch space.

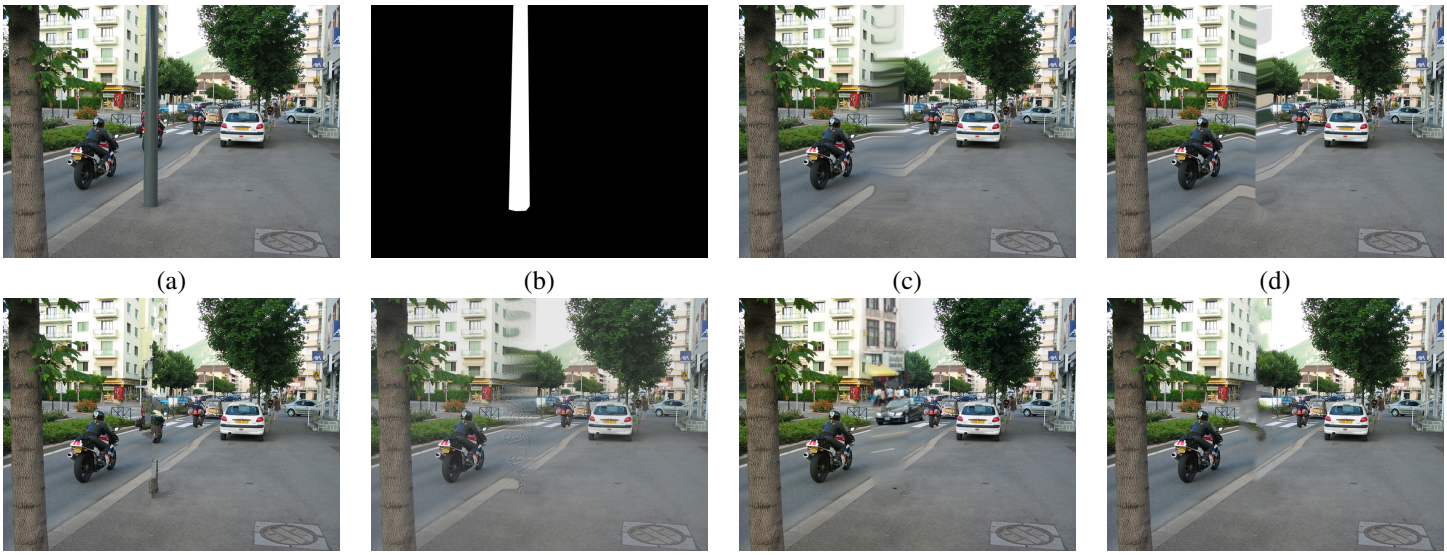
(a) (b) (c) (d) (e)

Fig. 15. Application to image editing. (a) Original image, from [36]. (b) Our inpainting result. (c) Compositing the kid over our result. (d) Result from [36]. (e) Result from [31].



(a) (b) (c)

Fig. 16. Application to image resizing. (a) Original image by Zanettco, sized 500×375. (b) In white the mask to be inpainted. (b) Resized 600×375.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 17. An example where none of these methods seems to work. (a) Original image. (b) In white the mask to be inpainted. (c) Result from Tschumperle [33]. (d) Result from Bornemann [24]. (e) Result from Criminisi et al. [7]. (f) Result from Bugeau et al. [34]. (g) Result from Hays et al. [35]. (h) Result obtained with our algorithm.

ACKNOWLEDGMENT

Major part of this work was developed for Mediapro inside the i3Media project which is partially funded by the Centro para el Desarollo Tecnolgico Industrial (CDTI), within the Ingenio 2010 initiative.

Aurélie Bugeau is supported by the Torres Quevedo program of the Ministerio de Educación y Ciencia in Spain.

Guillermo Sapiro is supported by NSF, NGA, ONR, ARO and DARPA.

REFERENCES

[1] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and grey levels," *IEEE Trans. Image Processing*, vol. 10, pp. 1200–1211, 2001.

[2] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, 2000.

[4] T. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions," *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436–449, December 2001.

[5] S. Masnou and J. M. Morel, "Level lines based disocclusion," in *IEEE ICIP (3)*, 1998, pp. 259–263.

[6] J. M. Ogden, E. H. Adelson, J. R. Bergen, and P. J. Burt, "Pyramid-based computer graphics," *RCA Engineer*, vol. 30(5), pp. 4–15, 1985.

[7] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[8] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *In Proceedings of the International Conference on Computer Vision*, 1999.

[9] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," in *International Conference on Computer Vision (ICCV)*, Oct 2003, Nice, France.

[10] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 29, pp. 1463–476, march 2007.

[11] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Processing*, vol. 17, pp. 53–69, 2008.

[12] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 882–889, 2003.

[13] L. Vese and S. Osher, "Modeling textures with total variation minimization and oscillating patterns in image processing," *Journal scientific Computing*, vol. 19, no. 1-3, pp. 553–572, 2003.

[14] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Non-local sparse models for image restoration," in *submitted*, 2009.

[15] H. Mobahi, S. Rao, and Y. Ma, "Data-driven image completion by image patch subspaces," in *Proceedings of the Picture Coding Symposium*, 2009.

[16] L. Demanet, B. Song, and T. Chan, "Image inpainting by correspondence maps: a deterministic approach," UCLA CAM R, Tech. Rep. 03-04, August 2003.

[17] M. Ashikhmin, "Synthesizing natural textures," *Proc. ACM Symp. Interactive 3D Graphics, ACM Press*, pp. 217–226, 2001.

[18] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, 2004.

[19] N. Komodakis and G. Tziritas, "Image completion using global optimization," in *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[20] M. Fadili, J.-L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *Computer Journal*, vol. 52, no. 1, pp. 64–79, 2009.

[21] M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, Segmentation, and Depth*. Springer-Verlag, Berlin, 1993.

[22] D. Tschumperle and R. Deriche, "Vector-valued image regularization with pdes: a common framework for different applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 506–517, April 2005.

[23] M. Bertalmio, "Strong-continuation, contrast-invariant inpainting with a third-order optimal pde," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 1934–1938, July 2006.

[24] F. Bornemann and T. März, "Fast image inpainting based on coherence transport," *Journal of Mathematical Imaging and Vision*, vol. 28, no. 3, pp. 259–278, 2007.

[25] G. Gilboa and S. Osher, "Nonlocal linear image regularization and supervised segmentation," *SIAM Multiscale Modeling and Simulation*, vol. 6, p. 595630, 2007.

[26] G. Peyré, "Manifold models for signals and images," *Computer Vision and Image Understanding*, vol. 113, pp. 249–260, 2009.

[27] J. Aujol, S. Ladjal, and S. Masnou, "Exemplar-based inpainting from a variational point of view," 2008.

[28] A. Efros and W. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, 2001.

[29] S. Shin, T. Nishita, and S. Shin, "On pixel-based texture synthesis by non-parametric sampling," *Computers & Graphics*, vol. 30, no. 5, pp. 767–778, oct 2006.

[30] L. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, 2000.

[31] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2009.

[32] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, 2003.

[33] D. Tschumperlé, "Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's," *International Journal of Computer Vision*, vol. 68, no. 1, pp. 65–82, 2006.

[34] A. Bugeau and M. Bertalmio, "Combining texture synthesis and diffusion for image inpainting," in *International Conference on Computer Vision Theory and Applications*, 2009.

[35] J. Hays and A. Efros, "Scene completion using millions of photographs," *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, vol. 26, no. 3, 2007.

[36] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman, "The patch transform and its applications to image editing," in *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.